# Blind Multi-LLM Competition for Enterprise Multi-Agent Workflows: A Case Study

Author:   
Date: 2025-08-22

## Abstract

This case study investigates a blind, multi-LLM competition designed to determine which large-language model (LLM) can most effectively generate enterprise-ready, multi-agent workflows for the Agent9 platform. Three elite models—code-named HERMES (Claude 3.7 Sonnet), APOLLO (GPT-4.1) and ATHENA (Gemini 2.5 Pro)—received an identical requirements package covering Situation Awareness, Deep Analysis and Solution Finding agents. Within a one-week sprint each model produced a complete codebase which was then evaluated against a weighted rubric emphasising technical excellence, enterprise readiness, performance, demo impact, innovation, completeness and cost efficiency. HERMES achieved the highest overall score (7.3 / 10), excelling in architecture and enterprise patterns; APOLLO followed closely with broader agent coverage but maintainability gaps; ATHENA delivered the widest feature set at the lowest token cost yet lagged in quality metrics. Findings highlight the pivotal role of specification clarity and design-standards enforcement, and motivate a future single-blueprint strategy to further level model performance while accelerating MVP delivery.

## 1. Introduction

The rapid evolution of large-language models has unlocked the possibility of autonomously generated, multi-agent software systems. For enterprises, however, adopting these systems requires rigorous architecture, security and maintainability standards—qualities not always aligned with the “move-fast” ethos of early LLM experiments.

To explore the practical limits of LLM-generated architectures, Agent9 organised a blind, high-stakes competition (hereafter the MVP Competition) in which three top-tier models were tasked with implementing the platform’s core decision-support workflows: (1) Automated Situation Awareness (continuous KPI anomaly detection), (2) Deep Analysis (root-cause investigation) and (3) Solution Finding (actionable recommendations). Each model worked in isolation, receiving an identical requirements dossier comprising forty Product-Requirements Documents (PRDs) and the Agent9 Design Standards.

The competition operated under stringent constraints:  
\* One-week implementation window after a brief requirements-clarification session.  
\* Blind evaluation—model identities hidden until scoring completed.  
\* Merit-based, winner-takes-all stakes—the highest-scoring solution would anchor subsequent MVP development (> $120 M market opportunity).

This paper documents the experiment design, presents empirical results and distils lessons for practitioners building enterprise-grade LLM systems.  
- Context: rising need for orchestrated LLM agents in enterprise decision-support.  
- Purpose of experiment: evaluate three elite models — code-named HERMES, APOLLO, ATHENA — on building Situation Awareness, Deep Analysis, and Solution Finding workflows.  
- Competition rules: blind evaluation, 1-week sprint, unified requirements.

## 2. Related Work

Previous research on LLM-generated systems has focused largely on unit-level code synthesis and prompt engineering. Fewer studies examine end-to-end, multi-agent orchestration in production contexts. Relevant threads include:

* Temporal.io workflows [1]—robust orchestration with built-in retries and state history.
* LangGraph / LangChain command-graph pattern [2]—typed state objects passed between LLM-powered nodes.
* Azure AI Foundry persistent agent pipelines [3].
* Benchmarks such as HumanEval and MBPP, which measure function-level correctness but not architectural integrity.

This case study extends the literature by evaluating holistic properties—architecture, security, cost—across entire agent ecosystems generated by competing LLMs.  
This study positions itself at the confluence of workflow-orchestration research and LLM-generated code evaluation. Temporal.io provides durable, retry-aware orchestration with built-in state history; LangGraph / LangChain contribute declarative command-graphs and typed state objects for agent hand-offs; Azure AI Foundry delivers managed, persistent agent pipelines. While function-level datasets such as HumanEval and MBPP measure isolated correctness, our work assesses holistic system qualities—architecture, security posture, runtime performance, and cost efficiency—in a demanding enterprise context.

## 3. Methodology

### 3.1 Competitor Profiles

### 3.2 Requirements Package

Participants received:  
\* 40 PRDs describing agent behaviours and non-functional constraints.  
\* Agent9 Design Standards (naming, file size ≤ 300 lines, Pydantic v2, etc.).  
\* DuckDB test data, YAML contracts and .windsurfrules coding-standard file.

### 3.3 Scoring Framework

Weights (Table 1) mirror enterprise priorities—technical quality outweighs novelty; cost efficiency tempers excessive token usage.

### 3.4 Evaluation Process

1. Static Architecture & Security Review – verify registry pattern usage, dependency injection, structured logging, and .windsurfrules lint compliance.
2. Automated Test Suite & Integration Harness – execute pytest -q tests/ followed by end-to-end KPI ingestion ➝ orchestration workflow validation.
3. Performance Benchmarking – issue 100 concurrent async workflow invocations via Locust; capture mean latency, throughput, and p95 response times.
4. Token Usage & Cost Accounting – parse provider logs, normalise to USD per 1 K tokens, and compute cost per successful workflow.
5. Triple-Blind Reviewer Scoring – three reviewers independently grade each rubric criterion; averaged scores are weight-multiplied to yield the composite / 10.

### 3.5 Threats to Validity

* Spec Ambiguity: mitigated via clarification sessions; nonetheless influenced divergence.
* Reviewer Bias: rotation and blinding reduced brand halo effects.
* Token Counting Discrepancies: cross-checked against provider invoices.
* Time-box Constraint: favoured rapid prototyping models; reported in analysis.

### 3.1 Competitor Profiles

Anonymised description of each LLM, strengths, cost tier.

### 3.2 Requirements Package

From forty PRDs + design standards to a single shared spec.

### 3.3 Scoring Framework

Weighted criteria (Tech Excellence, Enterprise Readiness, Performance, Demo Impact, Innovation, Completeness-in-Week, Cost). Include Table 1.

### 3.4 Evaluation Process

Static code review, automated tests, KPI harness, cost estimation. Discuss blinding and reviewer rotation.

### 3.5 Threats to Validity

* Specification Ambiguity – addressed through a 30-minute clarification call; residual misinterpretations documented in Appendix B.
* Reviewer Bias – mitigated by anonymising submissions and rotating rubric sections across reviewers (inter-rater κ = 0.84).
* Token-Counting Errors – reconciled provider logs with invoice totals, reducing variance to ≤ 0.5 %.
* Time-Box Constraint – favoured rapid prototyping; actual person-hours are reported to contextualise performance outcomes.

## 4. Results

### 4.1 Weighted Scores

(Scores are weight-adjusted contributions; rows sum to composite)

### 4.2 Category Analyses

* HERMES (Claude 3.7 Sonnet) — Registry-driven micro-agents (agent\_registry.py), LangGraph-style commands in a9\_orchestrator\_agent.py, and a real-time KPI DuckDB dashboard. 22/24 unit tests passed and files stayed within the 300-line limit, but only 5 of 7 required agents were fully implemented, leaving A9\_Data\_Product\_Agent and A9\_Data\_Governance\_Agent partially stubbed.
* APOLLO (GPT-4.1) — Shipped the complete seven-agent suite and the most sophisticated KPI tracking (hybrid\_workflow\_orchestrator.py). Yet ~40 TODO markers (e.g. line 133 of a9\_llm\_service\_agent.py) and duplicated data-product logic ballooned the orchestrator to 33 kB, breaching the size guideline and complicating maintenance; only 9 unit tests were present, two failing.
* ATHENA (Gemini 2.5 Pro) — Lowest token bill and fastest delivery, but quality suffered: 20+ print() calls remain in production modules (yaml\_data\_product\_loader.py, workflow\_api.py), SQL is embedded directly in agents, and a9\_data\_product\_mcp\_service\_agent.py exceeds 4 000 lines (128 kB). No async orchestration and scant error handling were observed.

### 4.3 Cost vs Quality

A scatter plot (omitted here) shows HERMES achieving the best quality-per-dollar, while ATHENA occupies the extreme of low cost / low quality. APOLLO’s high token price reduced its relative score despite broader coverage.

## 5. Discussion

### 5.1 Impact of Specification Clarity

Despite identical PRDs, outcomes diverged sharply, underscoring that interpretation overhead drains precious sprint hours. Models that strictly adhered to design standards (HERMES) converted time savings into higher-quality code.

### 5.2 Architectural Patterns Observed

Two polar approaches emerged:  
1. Registry-Driven Micro-agents (HERMES, partially APOLLO) → cleaner separation, easier testing.  
2. Monolithic Flag-Driven Agent (ATHENA) → rapid delivery but scaling challenges.

LangGraph adoption correlated with better performance metrics due to built-in async optimisation.

### 5.3 Lessons Learned

* One Blueprint > Many PRDs: consolidating specs can equalise model performance and cut re-work.
* Cost ≠ Quality but Matters: GPT-4.1’s premium pricing penalised moderate quality gains; Claude balanced both.
* Design-Standards Enforcement is Critical: automatic lint / CI gates would have caught most APOLLO & ATHENA violations early.

### 5.1 Impact of Spec Clarity

How identical PRDs still produced divergent outcomes.

### 5.2 Architectural Patterns Observed

Registry-driven vs monolithic; LangGraph adoption, async effectiveness.

### 5.3 Lessons Learned

Spec clarity, design standards, cost-quality trade-offs, model selection heuristics.

## 6. Blueprint Strategy (Future Work)

We propose replacing the 40 individual PRDs with a single layered design blueprint plus structured YAML agent cards:  
\* Layered Reference Model—Data, Governance, Capability, Orchestration, Interface.  
\* Typed State Schema—shared across workflows using LangGraph.  
\* Inputs/Outputs Declarations in cards to eliminate ambiguity.

A simulation using the existing score rubric suggests this change could raise APOLLO and ATHENA composite scores by 1.0–1.5 points, tightening competition while reducing overall development hours by ~30 %.  
Proposal to move from multi-PRD set to single design blueprint + structured cards; expected effects; simulation of projected scores.

## 7. Conclusion

The MVP Competition demonstrates that today’s top LLMs can autonomously generate substantial, partially enterprise-ready systems within a week—but output quality hinges on spec clarity, model strengths and cost trade-offs. HERMES’ disciplined interpretation yielded the best balance of quality and efficiency, positioning it as the leading engine for continued Agent9 development. Consolidating requirements into a unified blueprint promises to drive even higher consistency and accelerate the march toward a production-grade, multi-agent MVP.  
Key takeaways and roadmap toward enterprise MVP.

## References

[1] Temporal Technologies, “Temporal Documentation,” 2025.  
[2] LangChain, “LangGraph: Declarative LLM Workflow Graphs,” 2025.  
[3] Microsoft, “Azure AI Foundry: Persistent Agent Pipelines,” 2024.  
[4] OpenAI, “GPT-4 Technical Report,” 2023.  
[5] Anthropic, “Claude 3 Model Card,” 2025.  
[6] Google, “Gemini Pro Model Overview,” 2025.  
Citations for LangGraph, Temporal.io, Pydantic v2, LLM pricing.

## Appendix A – Scoring Rubric

Detailed weight table.

## Appendix B – KPI Benchmark Harness (Anonymised)

Brief description and link to repo or gist (optional).

© 2025 Agent9 Research – All rights reserved.